Generative Models for Physicists

Lei Wang*

Institute of Physics, Chinese Academy of Sciences
Beijing 100190, China

October 24, 2018

Abstract

Deep generative models are cutting-edge technologies which physicists can learn from deep learning. This note introduces the concept and principles of generative modeling, together with applications of modern generative models (autoregressive models, normalizing flows, variational autoencoders etc) as well as the old ones (Boltzmann machines) to physics problems. As a bonus, this note puts some emphasize on physics-inspired generative models which take insights from statistical, quantum, and fluid mechanics. The latest version of the note is at http://wangleiphy.github.io/. Please send comments, suggestions and corrections to the email address in below.

* wanglei@iphy.ac.cn

CONTENTS

1

GENERATIVE MODELING

What I can not create, I do not understand

Feynman on his blackboard

Indeed, having the ability to create something new is a good indication of deeper understanding, and higher level of intelligence. Generative modeling, by its name, it is about generating new instances with machine creativity. Generative models are fun! And, they are at the forefront of deep learning research [1].

1.1 PROBABILISTIC GENERATIVE MODELING

In the probabilistic language, generative modeling aims to learn the joint probability distribution of data and label $p(\boldsymbol{x}, y)$. With a generative model at hand, one can support the discriminative task through the Bayes formula $p(y|\boldsymbol{x}) = p(\boldsymbol{x}, y)/p(\boldsymbol{x})$, where $p(\boldsymbol{x}) = \sum_y p(\boldsymbol{x}, y)$. Moreover, one can generate new samples conditioned on its label $p(\boldsymbol{x}|y) = p(\boldsymbol{x}, y)/p(y)$. The generative models are also useful to support semi-supervised learning and reinforcement learning.

To wrap up, the goal of generative modeling is to *represent*, *learn* and *sample* from such high dimensional probability distributions. In the following, we will focus on learning $p(\boldsymbol{x})$. Learning joint probability distribution is similar.

Given an unlabelled dataset $\mathcal{D} = \{\boldsymbol{x}\}$, trying to model the joint probability is called *density estimation*. To see the objective function of such task, let us review a bit information theory. The Kullback-Leibler (KL) divergence reads

$$\mathbb{KL}(\pi||p) = \sum_{\boldsymbol{x}} \pi(\boldsymbol{x}) \ln \left[\frac{\pi(\boldsymbol{x})}{p(\boldsymbol{x})}\right], \tag{1}$$

which measures the distance between two probability distributions. We have $\mathbb{KL} \geq 0$ due to the Jensen inequality. The equality is achieved only when the two distributions are identity. The KL-divergence is not symmetric with respect to its arguments. $\mathbb{KL}(\pi||p)$ places high probability in p anywhere the data probability π is high, while $\mathbb{KL}(p||\pi)$ places low probability where the data probability π is low [2].

Info

The Jensen inequality [3] states that for convex $\smile$ functions f

$$\langle f(x)\rangle \geq f(\langle x\rangle). \tag{2}$$

Examples of convex $\smile$ functions $f(x) = x^2, e^x, e^{-x}, -\ln x, x\ln x$.

Introducing Shannon entropy $\mathbb{H}(\pi) = -\sum_{\boldsymbol{x}} \pi(\boldsymbol{x}) \ln \pi(\boldsymbol{x})$ and cross-entropy $\mathbb{H}(\pi, p) = -\sum_{\boldsymbol{x}} \pi(\boldsymbol{x}) \ln p(\boldsymbol{x})$, one sees that $\mathbb{KL}(\pi||p) = \mathbb{H}(\pi, p) - \mathbb{H}(\pi)$. Minimization of the KL-divergence is then equivalent to minimization of the cross-entropy since only it depends on the to-be-optimized parameters. In typical DL applications one only has i.i.d. samples from the target probability distribution $\pi(\boldsymbol{x})$, so one replaces it with the empirical estimation $\pi(\boldsymbol{x}) = \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}'\in\mathcal{D}} \delta(\boldsymbol{x} - \boldsymbol{x}')$. The cross entropy then turns out to be the negative log-likelihood (NLL) we met in the last chapter

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}} \ln[p_{\boldsymbol{\theta}}(\boldsymbol{x})]. \tag{3}$$

Idea 1: use a neural net to represent $p(\boldsymbol{x})$, but how to normalize ? how to sample ? Idea 2: use a neural net to transform simple prior $\boldsymbol{z}$ to complex data $\boldsymbol{x}$, but what is the likelihood ? How to actually learn the network?

Minimizing the NLL is a prominent (but not the only) way to train generative models, also known as Maximum Likelihood Estimation (MLE). The Eq. (3) appears to be a minor change compared to the discriminative task. However it causes huge challenges to change the conditional probability to probability function in the cost function. How to represent and learn such high dimensional probability distributions with the intractable normalization factor ? How could we marginalize and sample from such high dimensional probability distributions ? We will see that physicists have a lot to say about these problems since they love high dimensional probability, Monte Carlo methods and mean-field approaches. In fact, generative modeling has close relation to many problems in statistical and quantum physics, such as inverse statistical problems, modeling a quantum state and quantum state tomography.

Exercise 1 (Positivity of NLL). *Show that the NLL is positive for probability distributions of discrete variables. What about probability densities of continuous variables ?*

"Solving" means: 1. compute expected value of physical observables, 2. sampling configurations at various temperatures, 3. compute marginal probabilities given arbitrary subgroup of variables

For physicists, density estimation on data might not be the only thing we want to do. For example, in statistical physics one wants to solve the problem given bare energy function. Generative models can also help us on that with variational calculation by minimizing the reverse KL-divergence.

Info

Consider a statistical physics problem where $\pi(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/\mathcal{Z}$ and $\mathcal{Z} = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x})}$. We try to minimize the free energy $-\ln \mathcal{Z}$, which is unfortunately intractable in general. To proceed, we define a variational free energy

$$\mathcal{L} = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \ln \left[\frac{q(\boldsymbol{x})}{e^{-E(\boldsymbol{x})}} \right] = \langle E(\boldsymbol{x}) + \ln q(\boldsymbol{x}) \rangle_{z \sim q(\boldsymbol{x})} \tag{4}$$

for a normalized variational probability distribution $q(\boldsymbol{x})$. The two terms have the physical meaning of "energy" and "entropy" respectively. Crucially, since

$$\mathcal{L} + \ln \mathcal{Z} = \mathbb{KL}(q||\pi) \geq 0, \tag{5}$$

thus Eq. (4) is a variational upper bound of the physical free energy, $-\ln \mathcal{Z}$. The approximation becomes exact when the variational distribution approaches to the target probability. Equation (5) is known as Gibbs-Bogoliubov-Feynman inequality in physics.

The *density estimation* and *variational calculation* examples are by no means the only two applications of the generative models. Since they capture the whole distribution, by defining and adjusting the suitable cost function you can use generative model for creative applications, or enhance performance of downstream tasks. One thing we already see is that the requirement on the generative models are task dependent. For density estimation, we need to efficiently compute the likelihood (or at least its gradient given data). While for variational calculation, we need to be able to efficiently sample from the model as well.

1.2 GENERATIVE MODEL ZOO

This section we review several representative generative models. The key idea is to impose certain structural prior in the probability distribution. Each model has its own strengths and weakness. Exploring new approaches or combining the existing ones is an active research field in deep learning, with some ideas coming from physics.

1.2.1 *Boltzmann Machines*

As a prominent statistical physics inspired approach, the Boltzmann Machines (BM) model the probability as a Boltzmann distribution

$$p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{\mathcal{Z}}, \tag{6}$$

where $E(\boldsymbol{x})$ is an energy function and $\mathcal{Z}$, the partition function, is a normalization factor. The task of probabilistic modeling is then translated into designing and learning of the energy function to model observed data. For binary data, this is identical to the so called inverse Ising problem. Exploiting the maximum log-likelihood estimation, the gradient of Eq. (3) is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \left\langle \frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{x}\sim\mathcal{D}} - \left\langle \frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}. \tag{7}$$

The two terms are called positive and negative phase respectively. Intuitively, the positive phase tries to push down the energy of the observed data, therefore increases the model probability on the observed data. While the negative phase tries to push up the energy on samples drawn from the model, therefore to make the model probability more evenly distributed.

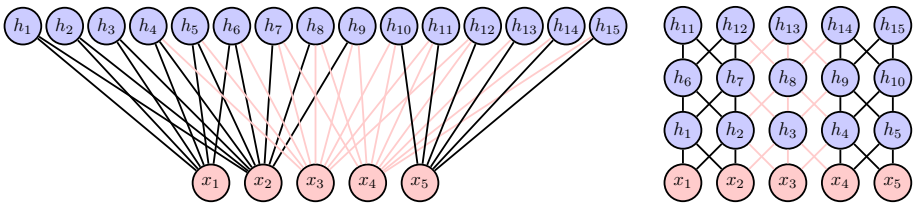
Example

Consider a concrete example of the energy model $E = -\frac{1}{2}W_{ij}x_i x_j$, the gradient Eq. (7) can be simply evaluated. And the gradient descent update

$$W_{ij} = W_{ij} + \eta \left(\langle x_i x_j \rangle_{\boldsymbol{x}\sim\mathcal{D}} - \langle x_i x_j \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})} \right). \tag{8}$$

The physical meaning of such update is quite appealing: one compares the correlation on the dataset and on the model, then strengthen or weaken the coupling accordingly.

The positive phase are quite straightforward to estimate by simply sampling batches from the dataset. While the negative phase typically involves the Markov chain Monte Carlo (MCMC) sampling. It can be very expensive to thermalize the Markov chain at each gradient evaluation step. The contrastive divergence (CD) algorithm [4] initialize the chain with a sample draw from the dataset and run the Markov chain only k steps. The reasoning is that if the BM has learned the probability well, then the model probability $p(\boldsymbol{x})$ resembles the one of the dataset anyway. Furthermore, the persistent CD [5] algorithm use the sample from last step to initialize the Monte Carlo chain. The



(a) RBM (b) DBM

Figure 1: RBM and DBM with the same number of neurons and connections. Information theoretical consideration shows that the DBM can potentially capture patterns that are impossible for the RBM [6].

logic being that in the gradient descent update to the model is small anyway, so accumulation of the Monte Carlo samples helps mixing. In practice, one run a batch of the Monte Carlo chain in parallel to estimate the expected value of the negative phase.

Exercise 2 (Mind the Gradient). *Define* $\Delta = \langle E(\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim\mathcal{D}} - \langle E(\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}$. *How is its gradient with respect to* $\boldsymbol{\theta}$ *related to Eq. (7) ?*

To increase the representational power of the model, one can introduce hidden variables in the energy function and marginalize them to obtain the model probability distribution

$$p(\boldsymbol{x}) = \frac{1}{\mathcal{Z}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}. \tag{9}$$

This is equivalent to say that $E(\boldsymbol{x}) = -\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$ in Eq. (6), which can be quite complex even for simple joint energy function $E(\boldsymbol{x},\boldsymbol{h})$. Differentiating the equation, we have

$$\frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}, \tag{10}$$

Therefore, in the presence of the hidden variables the gradient in Eq. (7) becomes

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \left\langle \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{h}\sim p(\boldsymbol{h}|\boldsymbol{x})} - \left\langle \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \right\rangle_{(\boldsymbol{x},\boldsymbol{h})\sim p(\boldsymbol{x},\boldsymbol{h})}, \tag{11}$$

which remains simple and elegant. However, the downside of introducing the hidden variables is that one needs even to perform expensive MCMC for the positive phase. An alternative approach is to use the mean-field approximation to evaluate these expectations approximately.

The restricted Boltzmann Machine (RBM) strives to have a balanced expressibility and learnability. The energy function reads

$$E(\boldsymbol{x},\boldsymbol{h}) = -\sum_i a_i x_i - \sum_j b_j h_j - \sum_{i,j} x_i W_{ij} h_j. \tag{12}$$

Since the RBM is defined on a bipartite graph shown in Fig. 1(a), its conditional probability distribution factorizes $p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j p(h_j|\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{h}) = \prod_i p(x_i|\boldsymbol{h})$, where

$$p(h_j = 1|\boldsymbol{x}) = \sigma\left(\sum_i x_i W_{ij} + b_j\right), \tag{13}$$

$$p(x_i = 1|\boldsymbol{h}) = \sigma\left(\sum_j W_{ij} h_j + a_i\right). \tag{14}$$

This means that given the visible units we can directly sample the hidden units in parallel, vice versa. Sampling back and force between the visible and hidden units is called block Gibbs sampling. Such sampling approach appears to be efficient, but it is not. The visible and hidden features tend to lock to each other for many steps in the sampling. In the end, the block Gibbs sampling is still a form of MCMC which in general suffers from long autecorrelation time and transition between modes.

Despite of appealing theory and historic importance, BM is now out of fashion in industrial applications due to limitations in its learning and sampling efficiency.

Info

For an RBM, one can actually trace out the hidden units in the Eq. (6) analytically and obtain

$$E(\boldsymbol{x}) = -\sum_i a_i x_i - \sum_j \ln(1 + e^{\sum_i x_i W_{ij} + b_j}). \tag{15}$$

This can be viewed as a Boltzmann Machine with fully visible units whose energy function has a softplus interaction. Using Eq. (7) and Eq. (15) one can directly obtain

$$-\frac{\partial \mathcal{L}}{\partial a_i} = \langle x_i \rangle_{\boldsymbol{x}\sim\mathcal{D}} - \langle x_i \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}, \tag{16}$$

$$-\frac{\partial \mathcal{L}}{\partial b_j} = \langle p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim\mathcal{D}} - \langle p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}, \tag{17}$$

$$-\frac{\partial \mathcal{L}}{\partial W_{ij}} = \langle x_i p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim\mathcal{D}} - \langle x_i p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}. \tag{18}$$

On see that the gradient information is related to the difference between correlations computed on the dataset and the model.

Exercise 3 (Improved Estimators). *To reconcile Eq. (11) and Eqs. (16-18), please convince yourself that* $\langle x_i p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim\mathcal{D}} = \langle x_i h_j \rangle_{\boldsymbol{x}\sim\mathcal{D}, \boldsymbol{h}\sim p(\boldsymbol{h}|\boldsymbol{x})}$ *and* $\langle x_i p(h_j = 1|\boldsymbol{x}) \rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})} = \langle x_i h_j \rangle_{(\boldsymbol{x},\boldsymbol{h})\sim p(\boldsymbol{x},\boldsymbol{h})}$. *The formal are improved estimators with reduced variances. In statistics this is known as the Rao-Blackwellization trick. Whenever you can perform marginalization analytically in a Monte Carlo calculation, please do it.*

Although in principle the RBM can represent any probability distribution given sufficiently large number of hidden neurons, the requirement can be exponential. To further increase the representational efficiency, one introduces the deep Boltzmann Machine (DBM) which has more than one layers of hidden neurons, see Fig. 1(b). Under information theoretical considerations, one can indeed show there are certain data which is impossible to represent using an RBM, but can possibly be represented by the DBM with the same number of hidden neurons and connections [6]. However, the downside of DBMs is that they are even harder to train and sample due the interactions among the hidden units [7].

1.2.2 *Autoregressive Models*

Arguably the simplest probabilistic model is the autoregressive models. They belong to the *fully visible Bayes network*. Basically, they breaks the full probability function into products of conditional probabilities, e.g.,

$$p(\boldsymbol{x}) = \prod_i p(x_i|\boldsymbol{x}_{<i}). \tag{19}$$

One can parameterize and learn the conditional probabilities using neural networks. In practice, one can model all these conditional probabilities using a single neural network, either a recurrent neural network with variable length, or using a feedforward neural network with masks. Note that these neural networks do not directly output the sample x_i, but the *parameters* of the conditional probability. For example, for continuous variables we can demand $p(x_i|\boldsymbol{x}_{<i}) = \mathcal{N}(x_i; \mu_i, \sigma_i^2)$, where the mean and variance are functions of $\boldsymbol{x}_{<i}$. The log-likelihood of a given data is easily computed as

$$\ln p(\boldsymbol{x}) = -\frac{1}{2}\sum_i \left(\left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 + \ln(2\pi\sigma_i) \right). \tag{20}$$

To sample from the autoregressive model, we can sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{1})$ and iterate the update rule

$$x_i = \sigma_i(\boldsymbol{x}_{<i})\epsilon_i + \mu_i(\boldsymbol{x}_{<i}). \tag{21}$$

Info

A slightly awkward but very enlightening way to compute the log-likelihood of the autoregressive model is to treat Eq. (21) as an invertible mapping between $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$, and invoke the probability transformation

$$\ln p(\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{1}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\epsilon}} \right) \right|. \tag{22}$$

Notice that Jacobian matrix is triangular, whose determinant can be easily computed to be Eq. (20). Generalizing this idea to more complex bijective transformations bring us to a general class of generative models called *Normalizing Flow* [8–16]. In particular, a stack of autoregressive transformations is called autoregressive flow (AF).

Despite their simplicity, autoregressive networks have achieved state of the art performances in computer vision (PixelCNN and PixelRNN [13]) and speech synthesis (WaveNet [14]). The downside of autoregressive models is that one has to impose an order of the conditional dependence which may not correspond to the global hierarchical structure of the data. Moreover, sequential sampling of the autoregressive model such as Eq. (21) is considered to be slow since they can not take advantage of modern hardware. Nevertheless, the generative process Eq. (21) is *direct* sampling, which is much more efficient compared to the Gibbs sampling of Boltzmann Machines.

Info

The inverse autoregressive flow (IAF) [12] changes the transformation Eq. (21) to be

$$x_i = \sigma_i(\boldsymbol{\epsilon}_{<i})\epsilon_i + \mu_i(\boldsymbol{\epsilon}_{<i}), \tag{23}$$

so that one can generate the data in parallel. The log-likelihood of the generated data also follows Eq. (20). However, the downside of the IAF is that it can not efficiently compute the likelihood of an arbitrary given data which is not generated by itself. Thus, IAF is not suitable for density estimation. IAF was originally introduced to improve the encoder of the VAE [12]. Recently, DeepMind use an IAF (Parallel WaveNet) [17] to learn the probability density of an autoregressive flow (WaveNet) [14], thus to improve the speech synthesis speed to meet the needs in real-world applications [18]. To train the parallel WaveNet, they minimize the *Probability Density Distillation* loss $\mathbb{KL}(p_{\text{IAF}}||p_{\text{AF}})$ [17] since it is easy to draw sample from IAF, and easy to compute likelihood of AF on given data.

1.2.3 *Normalizing Flow*

Normalizing flow is a family of bijective and differentiable (i.e., diffeomorphism) neural networks which maps between two continuous variables $\boldsymbol{z}$ and $\boldsymbol{x}$ of the same dimension. The idea is that the

physical variables can have more complex realistic probability density compared to the latent variables [8–16]

$$\ln p(\boldsymbol{x}) = \ln q(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|. \tag{24}$$

Since diffeomorphism forms a group, the transformation is compositional $\boldsymbol{x} = g(\boldsymbol{z}) = \cdots \circ g_2 \circ g_1(\boldsymbol{z})$, where each step is a diffeomorphism. And the log-Jacobian determinant in Eq. (24) is computed as $\ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right| = \sum_i \ln \left| \det \left(\frac{\partial g_{i+1}}{\partial g_i} \right) \right|$. To compute the log-likelihood of a given data, one first infer $\boldsymbol{z} = g^{-1}(\boldsymbol{x})$ and keep track of the log-Jacobian determinant in each step.

Bijectors are modular

The abstraction of a diffeomorphism neural network is called a bijector [19, 20]. Each bijector should provide interface to compute forward, inverse and log-Jacobian determinant in an efficient way. The bijectors can be assembled in a modular fashion to perform complex probability transformation. Because of their flexibility, they can act as drop in components of other generative models.

Example

As an example of Eq. (24), consider the famous Box-Muller transformation which maps a pair of uniform random variables $\boldsymbol{z}$ to Gaussian random variables $\boldsymbol{x}$

$$\begin{cases} x_1 = \sqrt{-2 \ln z_1} \cos(2\pi z_2), \\ x_2 = \sqrt{-2 \ln z_1} \sin(2\pi z_2). \end{cases} \tag{25}$$

Since $\left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right| = \left| \det \begin{pmatrix} \frac{-\cos(2\pi z_2)}{z_1 \sqrt{-2\pi \ln z_1}} & -2\pi \sqrt{-2 \ln z_1} \sin(2\pi z_2) \\ \frac{-\sin(2\pi z_2)}{z_1 \sqrt{-2\pi \ln z_1}} & 2\pi \sqrt{-2 \ln z_1} \cos(2\pi z_2) \end{pmatrix} \right| = \frac{2\pi}{z_1}$, we confirm that $q(\boldsymbol{x}) = p(\boldsymbol{z}) / \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right| = \frac{1}{2\pi} \exp\left(-\frac{1}{2}(x_1^2 + x_2^2) \right)$. Normalization flows are generalizations of this trick to higher dimensional spaces while still keepping the Jacobian determinants easy to compute.

We take the real-valued non-volume preserving transformation (Real NVP) [11] as an example of the normalizing flow. For each layer of the Real NVP network, we divide multi-dimensional variables $\boldsymbol{x}^\ell$ into two subgroups $\boldsymbol{x}^\ell = \boldsymbol{x}^\ell_< \cup \boldsymbol{x}^\ell_>$ and transform one subgroup conditioned on the other group at each step

$$\begin{cases} \boldsymbol{x}^\ell_< = \boldsymbol{x}^\ell_< \\ \boldsymbol{x}^\ell_> = \boldsymbol{x}^\ell_> \odot e^{s_\ell(\boldsymbol{x}^\ell_<)} + t_\ell(\boldsymbol{x}^\ell_<) \end{cases} \tag{26}$$

where $s_\ell(\cdot)$ and $t_\ell(\cdot)$ are two arbitrary functions (with correct input/output dimension) which we parametrize using neural networks.

It is clear that this transformation is easy to invert by reversing the scaling and translation operations. Moreover, the Jacobian determinant of the transformation is also easy to compute since the matrix is triangular. By applying a chain of these elementary transformations to various bipartitions one can transform in between a simple prior density and a complex target density. The Real NVP network can be trained with standard maximum likelihood estimation on data. After training, one can generate new samples directly by sampling latent variables according to the prior probability density and passing them through the network. Moreover, one can perform inference by passing the data backward through the network and obtain the latent variables. The log-probability of the data is efficiently computed as

$$\ln q(\boldsymbol{x}) = \ln p(\boldsymbol{z}) - \sum_{\ell,i} (s_\ell)_i, \tag{27}$$

where the summation over index i is for each component of the output of the s function.

From the above discussion, one sees that the crucial point of design flow-based generative model is to balance the expressibility and the computational effort of the Jacobian calculation. Typically, this is implemented via imposing internal structure of the network [8–16]. Recently, References shows that is is possible to take the "continuous-time" limit of the flow, which relax the structure constrain [21, 22]. Interestingly, the resulting generative model can be viewed as integrating an ordinary differential system for the data and its likelihood.

Monge-Ampère Flow

Reference [23] further revealed the optimal transport perspective of continuous-time normalizing flow and discussed means to impose the physical symmetries in the generating process. Consider latent variables $\boldsymbol{z}$ and physical variables $\boldsymbol{x}$ both living in $\mathbb{R}^N$. Given a diffeomorphism between them, $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{z})$, the probability densities in the latent and physical spaces are related by $p(\boldsymbol{z}) = q(\boldsymbol{x}) \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|$. The Brenier theorem [24] implies that instead of dealing with a multivariable generative map, one can consider a scalar valued generating function $\boldsymbol{x} = \nabla u(\boldsymbol{z})$, where the convex Brenier potential u satisfies the Monge-Ampère equation [25]

$$\frac{p(\boldsymbol{z})}{q(\nabla u(\boldsymbol{z}))} = \det \left(\frac{\partial^2 u}{\partial z_i \partial z_j} \right). \tag{28}$$

Given the densities p and q, solving the Monge-Ampère equation for u turns out to be challenging due to the nonlinearity in the determinant. Moreover, for typical machine learning and statistical physics problems, one faces an additional challenge that one does not even have direct access to both probability densities p and q. Instead, one only has independent and identically distributed samples from one of them, or one only knows one of the distributions up to a normalization constant. Therefore, solving the Brenier potential in these

contexts is a control problem instead of a boundary value problem. An additional computational challenge is that even for a given Brenier potential, the right-hand side of (28) involves the determinant of the Hessian matrix, which scales unfavorably as $\mathcal{O}(N^3)$ with the dimensionality of the problem.

To address these problems, we consider the Monge-Ampère equation in its *linearized form*, where the transformation is infinitesimal [26]. We write the convex Brenier potential as $u(\boldsymbol{z}) = ||\boldsymbol{z}||^2/2 + \epsilon\varphi(\boldsymbol{z})$, thus $\boldsymbol{x} - \boldsymbol{z} = \epsilon\nabla\varphi(\boldsymbol{z})$, and correspondingly $\ln q(\boldsymbol{x}) - \ln p(\boldsymbol{z}) = -\mathrm{Tr}\ln\left(I + \epsilon\frac{\partial^2\varphi}{\partial z_i \partial z_j}\right) = -\epsilon\nabla^2\varphi(\boldsymbol{z}) + \mathcal{O}(\epsilon^2)$. In the last equation we expand the logarithmic function and write the trace of a Hessian matrix as the Laplacian operator. Finally, taking the continuous-time limit $\epsilon \to 0$, we obtain

$$\frac{d\boldsymbol{x}}{dt} = \nabla\varphi(\boldsymbol{x}), \tag{29}$$

$$\frac{d\ln p(\boldsymbol{x}, t)}{dt} = -\nabla^2\varphi(\boldsymbol{x}), \tag{30}$$

such that $\boldsymbol{x}(0) = \boldsymbol{z}$, $p(\boldsymbol{x}, 0) = p(\boldsymbol{z})$, and $p(\boldsymbol{x}, T) = q(\boldsymbol{x})$, where t denotes continuous-time and T is a fixed time horizon. For simplicity here, we still keep the notion of $\boldsymbol{x}$, which now depends on time. The evolution of $\boldsymbol{x}$ from time $t = 0$ to T then defines our generative map.

The two ordinary differential equations (ODEs) compose a dynamical system, which describes the flow of continuous random variables and the associated probability densities under iterative change-of-variable transformation. One can interpret equations (29) and (30) as fluid mechanics equations in the Lagrangian formalism. Equation (29) describes the trajectory of fluid parcels under the velocity field given by the gradient of the potential function $\varphi(\boldsymbol{x})$. While the time derivative in (30) is understood as the "material derivative" [27], which describes the change of the local fluid density $p(\boldsymbol{x}, t)$ experienced by someone travels with the fluid.

The fluid mechanics interpretation is even more apparent if we write out the material derivative in (30) as $d/dt = \partial/\partial t + d\boldsymbol{x}/dt \cdot \nabla$, and use (29) to obtain

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot [p(\boldsymbol{x}, t)\boldsymbol{v}] = 0, \tag{31}$$

which is the continuity equation of a *compressible fluid* with density $p(\boldsymbol{x}, t)$ and velocity field $\boldsymbol{v} = \nabla\varphi(\boldsymbol{x})$. Obeying the continuity equation ensures that the flow conserves the total probability mass. Moreover, the velocity field is curl free $\nabla \times \boldsymbol{v} \equiv 0$ and the fluid follows a form of *gradient flow* [28]. The irrotational flow matches one's heuristic expectation that the flow-based generative model transports probability masses.

It should be stressed that although we use the optimal transport theory to motivate model architecture design, i.e., the gradient flow

for generative modeling, we do not have to employ the optimal transport objective functions for the control problem. The difference is that in generative modeling one typically fixes only one end of the probability density and aims at learning a suitable transformation to reach the other end. While for optimal transport one has both ends fixed and aims at minimizing the transportation cost. References [29–31] adapted the Wasserstein distances in the optimal transport theory as an objective function for generative modeling.

1.2.4 *Variational Autoencoders*

One of the creators of VAE, Max Welling, did his PhD on gravity theory under the supervision of 't Hooft in late 90s.

Variational autoencoder (VAE) is an elegant framework for performing variational inference [32], which also has deep connection variational mean field approaches in statistical physics. In fact, the predecessor of VAE is called Helmholtz machines [33]. The general idea of an autoencoder is to let the input data go through a network with bottleneck and restore itself. After training, the first half of the network is an encoder which transform the data $\boldsymbol{x}$ into the latent space $\boldsymbol{z}$. And the second half of the network is a decoder which transform latent variables into the data manifold. The bottleneck means that we typically require that the latent space has lower dimension or simpler probability distribution than the original data.

Intractable posterior

Suppose the latent variables $p(\boldsymbol{z})$ follow a simple prior distribution, such as an independent Gaussian. The decoder is parameterized by a neural network which gives the conditional probability $p(\boldsymbol{x}|\boldsymbol{z})$. Thus, the joint probability distribution of the visible and latent variables is also known $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$. However, the encoder probability given by the posterior $p(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{x}, \boldsymbol{z})/p(\boldsymbol{x})$ is much more difficult to evaluate since normalization factor $p(\boldsymbol{x})$ is intractable. One needs to marginalize the latent variables $\boldsymbol{z}$ in the joint probability distribution $p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z})\mathrm{d}\boldsymbol{z}$.

The intractable integration over the latent variables also prevent us minimizing the NLL on the dataset. To deal with such problem, we employ variational approach originated from statistical physics. The variational Bayes methods is a an application of the variational free energy calculation in statistical physics Eq. (5) for inference problem. For each data we introduce

$$\mathcal{L}(\boldsymbol{x}) = \langle -\ln p(\boldsymbol{x}, \boldsymbol{z}) + \ln q(\boldsymbol{z}|\boldsymbol{x})\rangle_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}, \tag{32}$$

which is a variational upper bound of $-\ln p(\boldsymbol{x})$ since $\mathcal{L}(\boldsymbol{x}) + \ln p(\boldsymbol{x}) = \mathbb{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) \geq 0$. We see that $q(\boldsymbol{z}|\boldsymbol{x})$ provides a variational approximation of the posterior $p(\boldsymbol{z}|\boldsymbol{x})$. By minimizing $\mathcal{L}$ one effectively pushes the two distributions together. And the variational free energy becomes exact only when $q(\boldsymbol{z}|\boldsymbol{x})$ matches to $p(\boldsymbol{z}|\boldsymbol{x})$. In fact, $-\mathcal{L}$ is called evidence lower bound (ELBO) in variational inference.

This breakup is also the foundation of the Expectation-Maximization algorithm, where one iterates alternatively between optimizing the variational posterior (E) and the parameters (M) to learn models with latent variables [34].

We can obtain an alternative form of the variational free energy

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = -\left\langle \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right\rangle_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} + \mathbb{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})). \tag{33}$$

The first term of Eq. (33) is the reconstruction negative log-likelihood, while the second term is the KL divergence between the approximate posterior distribution and the latent prior. We also be explicit about the network parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$ of the encoder and decoder.

The decoder neural network $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ accepts the latent vector $\boldsymbol{z}$ and outputs the parametrization of the conditional probability. It can be

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \sum_i x_i \ln \hat{x}_i + (1-x_i)\ln(1-\hat{x}_i), \tag{34}$$

$$\hat{\boldsymbol{x}} = \texttt{DecoderNeuralNet}_{\boldsymbol{\theta}}(\boldsymbol{z}), \tag{35}$$

for binary data. And

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \ln \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\sigma}^2\mathbf{1}), \tag{36}$$

$$(\boldsymbol{\mu},\boldsymbol{\sigma}) = \texttt{DecoderNeuralNet}_{\boldsymbol{\theta}}(\boldsymbol{z}), \tag{37}$$

for continuous data. Gradient of Eq. (33) with respect to $\boldsymbol{\theta}$ only depends on the first term.

Similarly, the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is also parametrized as a neural network. To optimize $\boldsymbol{\phi}$ we need to compute the gradient with respect to the sampling process, which we invoke the *reparametrization trick*. To generate sample $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ we first sample from an independent random source, say $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{1})$ and pass it through an invertible and differentiable transformation $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\epsilon})$. The probability distribution of the encoder is related to the one of the random source by

$$\ln q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{1}) - \ln \left|\det\left(\frac{\partial g_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}}\right)\right|. \tag{38}$$

Suppose that the log-determinant is easy to compute so we can sample the latent vector $\boldsymbol{z}$ given the visible variable $\boldsymbol{x}$ and an independent random source $\boldsymbol{\epsilon}$. Now that the gradient can easily pass through the sampling process

$$\nabla_{\boldsymbol{\phi}} \left\langle f(\boldsymbol{x},\boldsymbol{z})\right\rangle_{\boldsymbol{z}\sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})} = \left\langle \nabla_{\boldsymbol{\phi}} f(\boldsymbol{x}, g_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\epsilon}))\right\rangle_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{1})}. \tag{39}$$

Info

As an alternate, the REINFORCE [35] (score function) estimator of the gradient reads

$$\nabla_{\boldsymbol{\phi}} \left\langle f(\boldsymbol{x},\boldsymbol{z})\right\rangle_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} = \left\langle f(\boldsymbol{x},\boldsymbol{z})\nabla_{\boldsymbol{\phi}} \ln q_{\phi}(\boldsymbol{z}|\boldsymbol{x})\right\rangle_{\boldsymbol{z}\sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}. \tag{40}$$

Compared to the reparametrization Eq. (39) REINFORCE usually has larger variance because it only uses the scalar function $\ln q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ instead of the vector information of the gradient $\nabla_{\phi} f(\boldsymbol{x},\boldsymbol{z})$. An ad-

vantage of REINFORCE is that it can also work with discrete latent variables. See Ref. [36] for the research frontier for low variance unbiased gradient estimation for discrete latent variables.

Suppose each component of the latent vector follows independent Gaussian whose mean and variance are determined by the data $\boldsymbol{x}$, we have

$$\ln q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \sigma^2 \mathbf{1}), \tag{41}$$

$$(\boldsymbol{\mu}, \sigma) = \texttt{EncoderNeuralNet}_{\boldsymbol{\phi}}(\boldsymbol{x}). \tag{42}$$

And the way to sample the latent variable is

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{1}), \tag{43}$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \sigma \odot \boldsymbol{\epsilon}. \tag{44}$$

The KL term in Eq. (33) can be evaluated analytically [32] in this case.

After training of the VAE, we obtain an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ and a decoder $p(\boldsymbol{x}|\boldsymbol{z})$. The encoder performs dimensionality reduction from the physical space into the latent space. Very often, different dimensions in the latent space acquire semantic meaning. By perform arithmetic operations in the latent space one can interpolate between physical data. Optimization of chemical properties can also be done in the low dimensional continuous latent space. The decoder is a generative model, which maps latent variable into the physical variable with rich distribution.

Info

The marginal NLL of the VAE can be estimated using importance sampling

$$-\ln p(\boldsymbol{x}) = -\ln \left\langle \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})} \right\rangle_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})}. \tag{45}$$

By using the Jensen's inequality (2) one can also see that the variational free energy Eq. (32) is an upper bound of Eq. (45).

1.2.5 *Tensor Networks*

A new addition to the family of generative models is the tensor network state. In a quantum inspired approach one models the probability as the wavefunction square

$$p(\boldsymbol{x}) = \frac{|\Psi(\boldsymbol{x})|^2}{\mathcal{Z}}, \tag{46}$$

where $\mathcal{Z}$ is the normalization factor. This representation, named as Born Machine [37], transforms many approaches of representing

quantum state into machine learning. Consider binary data, we can represent wavefunction using the matrix product state (MPS) [38]

$$\Psi(\boldsymbol{x}) = \mathrm{Tr}\left(\prod_i A^i[x_i]\right). \tag{47}$$

The size of each matrix is called the bond dimension of the MPS representation. They control the expressibility of the MPS parameterization. The MPS can be learned using maximum likelihood estimation as before. Although other loss functions such as fidelity of quantum states can also be considered [39, 40].

Adaptive learning

An advantage of using MPS for generative modeling is that one can adopt algorithms developed for quantum many-body states such as the DMRG for parameter learning. For example, one can perform "two-site" optimization by merging two adjacent matrices together and optimizing its tensor elements. After the optimization the rank of the two site tensor may grow, one can thus dynamically adjust the bond dimension of the MPS representation during learning. As a consequence, the expressibility of the model grows as it observes the data, which is different from conventional generative models with fixed network with fixed number of parameters.

Efficient gradient

Another advantage of MPS as a generative model is that the gradient of the NLL (3) can be computed efficiently

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = -2\left\langle \frac{\partial \ln \Psi(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{x}\sim\mathcal{D}} + 2\left\langle \frac{\partial \ln \Psi(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}. \tag{48}$$

Note that the negative phase (second term) can also be written as $\mathcal{Z}'/\mathcal{Z}$, where $\mathcal{Z}' = 2\sum_{\boldsymbol{x}} \Psi'(\boldsymbol{x})\Psi(\boldsymbol{x})$ and the prime means derivatives with respect to the network parameter $\boldsymbol{\theta}$. Crucially, for MPS both $\mathcal{Z}'$ and $\mathcal{Z}$ can be evaluated efficiently via tensor contractions. So the gradient can be computed efficiently without resorting to the contrastive divergence, in contrast to the Boltzmann Machines (7). The NLL is also tractable so that MPS model knows the normalized density of each sample.

Direct sampling

Finally, tractable normalization factor of MPS allows one to perform *direct* sampling instead of using MCMC used in the Boltzmann Machines. While compared to the autoregressive models, one can perform data restoration by removing any part of the data. This is because tensor networks expresses an undirected (instead of directed) probability dependence fo the data.

These aforementioned advantages apply as well to other unitary tensor networks such as the tree tensor network and MERA. It is yet to been seen whether one can unlock the potential of tensor networks for real world AI applications. Using Eq. (46) and associated quantum-inspired approaches (or even a quantum device) provide a great chance to model complex probabilities. While on a more conceptual level, one wish to have have more quantitative and interpretable

approaches inspired by quantum physics research. For example, Born Machine may give us more principled structure designing and learning strategies for modeling complex dataset, and provide a novel theoretical understandings of the expressibility of generative models the quantum information perspective.

There are at least two motivations of using the tensor networks for machine learning. First, tensor network and algorithms provide principled approaches for discriminative and generative tasks with possibly stronger representational power. In fact, the mathematical structure of tensor network states and quantum mechanics appear naturally when one tries to extend the probabilistic graphical models while still attempt to ensure the positivity of the probability density [41, 42]. Second, tensor networks are doorways to quantum machine learning because many of the tensor networks are formally equivalent to quantum circuits. Tensor network states provide means of architecture design and parameter initialization for quantum circuits [43]. By now, tensor networks have certainly caught attentions of some of the machine learning practitioners [44, 45]. However, we are still awaiting for an event similar to AlexNet, where the tensor network machine learning approach wins over the traditional approaches by a large margin. With accumulations of the results and techniques, this is likely to happen in the coming years, at least in one specific application domain.

[LW: Not only architectures, but also]

1.2.6 *Generative Adversarial Networks*

Different from the generative models introduced till now, the Generative Adversarial Networks (GAN) belong to the *implicit* generative models. That is to say that although one can generate samples using GAN, one does not have direct access to its likelihood. So obviously training of GAN is also not based on maximum likelihood estimation.

A generator network maps random variables $\boldsymbol{z}$ to physical data $\boldsymbol{x}$. A discriminator network D is a binary classifier which tries tell whether the sample is from the dataset $\mathcal{D}$ (1) or synthesized (0). On the expanded dataset $\{(\boldsymbol{x}, 1), (G(\boldsymbol{z}), 0)\}$, the cross-entropy cost reads

$$\mathcal{L} = -\left\langle \ln D(\boldsymbol{x}) \right\rangle_{\boldsymbol{x}\sim\mathcal{D}} - \left\langle \ln\left(1 - D(G(\boldsymbol{z}))\right) \right\rangle_{\boldsymbol{z}\sim p(\boldsymbol{z})} . \tag{49}$$

Such cost function defines a minimax game $\max_G \min_D \mathcal{L}$ between the generator and the discriminator, where the generator tries to forge data to confuse the discriminator.

Since the loss function does not involve the probability of the generated samples, one can use an arbitrary neural network as the generator. Giving up likelihood increases the flexibility of the generator network at the cost that it is harder to train and evaluate. Assess the performance of GAN in practice often boils down to beauty contest.

Lacking an explicit likelihood function also limits its applications to physics problems where quantitative results are important.

Recall the Born Machine mentioned in Sec. 1.2.5, suppose one implements a Born Machine using a quantum circuit, the resulting model would be an implicit model [46]. Since one usually does not have direct access to the quantum state of an actual quantum state, adversarial training against a classical neural network can be a way to learn the quantum circuit as a probabilisitic generative model [47].

Table 1: A summary of generative models and their salient features. Question marks mean generalizations are possible, but nontrivial.

Name	Training Cost	Data Space	Latent Space	Architecture	Sampling	Likelihood	Expressibility	Difficulty (Learn/Sample)
RBM	Log-likelihood	Arbitrary	Arbitrary	Bipartite	MCMC	Intractable partition function	★	☠☠/☠☠☠
DBM	ELBO	Arbitrary	Arbitrary	Bipartite	MCMC	Intractable partition function & posterior	★★★	☠☠/☠☠☠
Autoregressive Model	Log-likelihood	Arbitrary	None	Ordering	Sequential	Tractable	★★	☠/☠☠
Normalizing Flow	Log-likelihood	Continuous	Continuous, Same dimension as data	Bijector	Parallel	Tractable	★★	☠/☠
VAE	ELBO	Arbitrary	Continuous	Arbitrary?	Parallel	Intractable posterior	★★★	☠/☠
MPS/TTN	Log-likelihood	Arbitrary?	None or tree tensor	No loop	Sequential	Tractable	★★★	☠☠/☠☠
GAN	Adversarial	Continuous	Arbitrary?	Arbitrary	Parallel	Implicit	★★★★	☠☠☠/☠
Quantum Circuit	Adversarial	Discrete	Discrete	Arbitrary	Parallel	Implicit	★★★★★	☠☠☠☠/☠

1.3 SUMMARY

In the discussions of generative models we have touched upon a field called probabilistic graphical models [48]. They represent independence relation using graphical notations. The graphical models with undirected edges are called Markov random field, which can be understood as statistical physics models (Sec. 1.2.1). Typically, it is hard to sample from a Markov random field unless it has a tree structure. While the graphical models with directed edges are called Bayes network, which describe conditional probability distribution (Sec. 1.2.2). The conditional probabilities allows ancestral sampling which start from the root node and follow the conditional probabilities.

As we have seen, feedforward neural networks can be used as key components for generative modeling. They transform the probability distribution of the input data to certain target probability distribution. Please be aware that there are subtle differences in the interpretations of these neural nets' outputs. They can either be parametrization of the conditional probability $p(\boldsymbol{x}|\boldsymbol{z})$ (Secs. 1.2.2, 1.2.4) or be the samples $\boldsymbol{x}$ themselves (Secs. 1.2.3, 1.2.6). Table 1 summarized and compared the main features of various generative models discussed in this note.

In fact, various models introduce in this section is also related. Seeking their relation or trying to unify them provides one a deeper understanding on generative modeling. First of all, the Boltzmann Machines, and in general probabilistic graphical models, are likely to be closely related to the tensor networks. In particular cases, the exact mappings between RBM and tensor networks has been worked out [6]. In general, it is still rewarding to explore the connections of representation and learning algorithms between the two classes of models. Second, the autoregressive models are closely related to the normalizing flows viewed as a transformation of probability densities. While in [8] it was even argued on the connections to the variational autoencoder. Finally, combining models to take advantage of both worlds is also a rewarding direction [10, 12, 49].

2

PHYSICS APPLICATIONS

We arrange the topics according to the domain of applications, but not on the specific models.

2.1 VARIATIONAL ANSATZ

Reference [50] obtained excellent variational energy for non-frustrated quantum spin systems by adopting the Restricted Boltzmann Machines in Sec. 1.2.1 as a variational ansatz. The ansatz can be viewed as an alternative of Jastrows. But it is more flexible in the sense that it encodes multi-body correlations in an efficient way.

Later studies [6, 51, 52] connect the RBM variational ansatz to tensor network states. References [53, 54] analyzed their expressibility from quantum entanglement and computational complexity points of view respectively. Out of these works, one sees that the neural network states can be advantageous for describing highly entangled quantum states, and models with long range interactions. Another particular interesting application is on the chiral topological states, in which the standard PEPS ansatz suffer from fundamental difficulties [52, 55].

Another interesting direction is to interpret that RBM, in particular, the one used in [50] as shallow convolutional neural networks. Along this line, it is natural to go systematically to deeper neural networks and employ deep learning frameworks for automatic differentiation in the VMC calculation [56]. Reference [57] carried out the VMC calculation for small molecules in the first quantization formalism, in which the antisymmetric property of the wavefunction was taken with special care.

It appears to the author that further development calls for innovations in the the optimization scheme which is beyond the wavefunction ansatz, e.g. direct generative sampling, and low variance gradient estimator. As a one step towards this goal, the authors employ variational autoregressive networks to study Ising and spin glass problems [58]. This calculation follows a variational free energy framework, which generalize the celebrated Weiss and Bethe mean field theories.

2.2 RENORMALIZATION GROUP

Renormalization Group (RG) is a fundamental concept in theoretical physics. In essence, RG keeps relevant information while reducing the dimensionality of data. The connection of RG and deep learning is quite intriguing since on one hand side it brings deep learning machineries into solving physical problems with RG, and on the other hand site, it may provide theoretical understanding to deep learning.

References [59] proposed a generative Bayesian network with a MERA inspired structure. Reference [60] connects the Boltzmann Machines with decimation transformation in real-space RG. Reference [61] connects principal component analysis with momentum shell RG. Reference [62] proposed to use mutual information as a criteria for restoring the RG behavior in the training of Boltzmann Machines. Lastly, Reference [63] proposed a variational RG framework by stacking the bijectors (Sec. 1.2.3) into a MERA-liked structure. The approach provides a way to identify collective variables and and their effective interaction. The collective variables in the latent space has reduced mutual information. They can be regarded as nonlinear and adaptive generalizations of wavelets. Training of the NeuralRG network employs the probability density distillation (Sec. 1.2.2) on the bare energy function, in which the training loss provides a variational upper bound of the physical free energy. The NeuralRG approach implements an information preserving RG procedure, which is useful for exploring holographic mapping [64].

2.3 MONTE CARLO UPDATE PROPOSALS

Markov chain Monte Carlo (MCMC) finds wide applications in physics and machine learning. Since the major drawback of MCMC compared to other approximate methods is its efficiency, there is a strong motivation to accelerate MCMC simulations within both physics and machine learning community. Loosely speaking, there are at least three ideas of accelerating Monte Carlo sampling using machine learning techniques.

First, Reference [65] trained surrogate functions to speed up hybrid Monte Carlo simulation [66] for Bayesian statistics. The surrogate function approximates the potential energy surface of the target problem and provides an easy way to compute derivatives. Recently, there were papers reporting similar ideas for physics problems. Here, the surrogate model can be physical models such as the Ising model [67] or molecular gases [68], or general probabilistic graphical models such as the restricted Boltzmann machine [69]. For Monte Carlo simulations involving fermion determinants [68, 70] the approach is more profitable since the updates of the original model is much heavier than the surrogate model. However, the actual benefit depends

on the particular problem and the surrogate model. A drawback of these surrogate function approaches is that they require training data to start with, which is known as the "cold start" problem in analog to the recommender systems [68]. Using the adaptive approach of [71] one may somewhat alleviate this "chicken-egg" problem.

Second, there were more recent attempts in machine learning community trying to directly optimize the proposal probability via reinforcement learning [72–74]. These papers directly parameterize the proposal probability as neural networks and optimize objective functions related to the efficiency, e.g., the difference of proposals such as the squared jump distances. To ensure unbiased physical results, it is crucial to keep track of the proposal probability of an update and its reverse move for the detailed balance condition. Both A-NICE-MC [72] and L2HMC [73] adopted normalizing flows (Sec. 1.2.3). The later paper is particularly interesting because it systematically generalizes the celebrated hybrid Monte Carlo [66] to a learnable framework. Reference [75] used reinforcement learning to devise updates for frustrated spin models. In fact, besides the efficiency boost one can aim at algorithmic innovations in the Monte Carlo updates. Devising novel update strategies which can reduce the auto correlation between samples was considered to be the art MCMC methods. An attempt along this line is Ref. [76], which connected the Swendsen-Wang cluster and the Boltzmann Machines and explored a few new cluster updates.

Lastly, the NeuralRG technique [63] provides another approach to learn Monte Carlo proposal without data. Since the mapping to the latent space reduces the complexity of the distribution, one can perform highly efficient (hybrid) Monte Carlo updates in the latent space and obtain unbiased physical results. This can be regarded as an extension of the Fourier space or wavelets basis Monte Carlo methods, except that now the representation is learned in an adaptive fashion.

By the way, to obtain unbiased physical results one typically ensures detailed balance condition using Metropolis-Hastings acceptance rule. Thus, one should employ generative models with *explicit* and *tractable* densities for update proposals. This rules out GAN and VAE in the game, at least for the moment.

2.4 CHEMICAL AND MATERIAL DESIGN

A recent example in chemistry design is to use the VAE to map string representation of molecules to a continuous latent space and then perform differential optimization for desired molecular properties [77]. Like many deep learning applications in natural language and images, the model learned meaningful low dimensional representation in the latent space. Arithmetics operations have physical

(or rather chemical) meanings. There were also attempts of using GANs for generating molecules. See [78] for a recent review.

2.5 QUANTUM INFORMATION SCIENCE AND BEYOND

The Restricted Boltzmann Machines were used to decode quantum error [79], for quantum state tomography [80], and for detecting Bell nonlocality in quantum mechanics [81].

In general, generative model might be one of the killer application of the quantum machine learning [82, 83]. Building on the probabilistic interpretation of quantum mechanics, one can envision a quantum generative model [84]. It expresses the probability distribution of a dataset as the probability associated with the wavefunction. The theory behind it is that even measured on a fixed bases, the quantum circuit can express probability distribution that are intractable to classical computers. In particular, there was a demonstration [85] on generative machine learning. References [46, 86] explores the practical aspects of scaling up the generative learning on actual quantum devices, such as objective function and gradient based training.

3

RESOURCES

Slides and Reviews

- Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks
- Ruslan Salakhutdinov, Learning Deep Generative Models
- Shakir Mohamed and Danilo Rezende, Deep Generative Models
- Ryan Adams, A Tutorial on Deep Probabilistic Generative Models
- David Duvenaud, Generative Models

Blogs

- Rui Shu, Change of Variables: A Precursor to Normalizing Flow
- Eric Jang, Normalizing Flows Tutorial I, II
- OpenAI, Glow: Better Reversible Generative Models
- DeepMind, WaveNet: A Generative Model for Raw Audio, High-fidelity speech synthesis with WaveNet

Tutorial Codes

- MADE (Masked Autoencoder Density Estimation)
- PixelCNN
- RBM for image restoration
- Variational Autoregressive Network
- Neural Renormalization Group (with Real NVP layers)
- Continuous-time Normalizing Flow
- MPS for generative modeling

ACKNOWLEDGEMENTS

We thank fruitful collaborations and discussions with Pan Zhang, Yehua Liu, Dong-Ling Deng, Zhi-Yuan Xie, Linfeng Zhang, Weinan E, Jing Chen, Song Cheng, Haidong Xie, Tao Xiang, Zhao-Yu Han, Jun Wang, Dian Wu, Xiu-Zhe Roger Luo, Xun Gao, Zi Cai, Li Huang, Yi-feng Yang, Yang Qi, Junwei Liu, Ziyang Meng, Yi-Zhuang You, Miles Stoudenmire, Giuseppe Carleo, Matthias Rupp, Alejandro Perdomo-Ortiz. The authors are supported by the National Natural Science Foundation of China under Grant No. 11774398.

BIBLIOGRAPHY

[1] Andrej Karpathy, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian GoodFellow, Durk Kingma, Jonathan Ho, Rein Houthooft, Tim Salimans, John Schulman, Ilya Sutskever, and Wojciech Zaremba. OpenAI Post on Generative Models, 2016. URL https://blog.openai.com/generative-models/.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org/.

[3] David J C MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[4] Geoffrey E Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Comput.*, 14:1771, 2002.

[5] Tijmen Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *Proceedings of teh 25th international conference on Machine Learning*, page 1064, 2008.

[6] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. On the Equivalence of Restricted Boltzmann Machines and Tensor Network States. *Phys. Rev. B*, 97:085104, 2018. doi: 10.1103/PhysRevB.97.085104. URL http://arxiv.org/abs/1701.04831.

[7] Ruslan Salakhutdinov. Learning Deep Generative Models. *Annu. Rev. Stat. Its Appl.*, 2:361–385, 2015. doi: 10.1146/annurev-statistics-010814-020120. URL http://www.annualreviews.org/doi/10.1146/annurev-statistics-010814-020120.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Nonlinear Independent Components Estimation. *arXiv*, 2014. doi: 1410.8516. URL http://arxiv.org/abs/1410.8516.

[9] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. *arXiv*, 2015. URL http://arxiv.org/abs/1502.03509.

[10] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *arXiv*, 2015. URL http://arxiv.org/abs/1505.05770.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv*, 2016. doi: 1605.08803. URL http://arxiv.org/abs/1605.08803.

[12] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv*, 2016. URL http://arxiv.org/abs/1606.04934.

[13] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In *Int. Conf. Mach. Learn. I(CML)*, volume 48, pages 1747—-1756, 2016. URL https://arxiv.org/pdf/1601.06759.pdf{%}0Ahttp://arxiv.org/abs/1601.06759.

[14] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv*, 2016. URL http://arxiv.org/abs/1609.03499.

[15] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv*, 2017. URL http://arxiv.org/abs/1705.07057.

[16] D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *ArXiv e-prints*, July 2018.

[17] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv*, 2017. URL http://arxiv.org/abs/1711.10433.

[18] Aaron van den Oord, Yazhe Li, and Igor Babuschkin. High-fidelity speech synthesis with WaveNet, 2017. URL https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/.

[19] Pyro Developers. Pyro, 2017. URL http://pyro.ai/.

[20] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow Distributions. *arXiv*, 2017. URL http://arxiv.org/abs/1711.10604.

[21] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. *ArXiv e-prints*, June 2018.

[22] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *ArXiv e-prints*, October 2018.

[23] L. Zhang, W. E, and L. Wang. Monge-Amp\'ere Flow for Generative Modeling. *ArXiv e-prints*, September 2018.

[24] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[25] Luis A Caffarelli, NSF-CBMS Conference on the Monge Ampère Equation: Applications to Geometry, Optimization, and Mario Milman. *Monge Ampere equation: applications to geometry and optimization*. American Mathematical Society, 1998.

[26] Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

[27] Kip S Thorne and Roger D Blandford. *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*. Princeton University Press, 2017.

[28] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[29] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

[30] Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. From optimal transport to generative modeling: the VEGAN cookbook. 2017. URL http://arxiv.org/abs/1705.07642.

[31] Aude Genevay, Gabriel Peyré, and Marco Cuturi. GAN and VAE from an Optimal Transport Point of View. 2017. URL http://arxiv.org/abs/1706.01807.

[32] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. 2013. URL http://arxiv.org/abs/1312.6114.

[33] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The Helmholtz Machine. 904:889–904, 1995.

[34] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[35] Ronald J. Wiliams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1023/A:1022672621406.

[36] George Tucker, Andriy Mnih, Chris J. Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv*, 2017. URL http://arxiv.org/abs/1703.07370.

[37] Song Cheng, Jing Chen, and Lei Wang. Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines. *arXiv*, 2017. URL http://arxiv.org/abs/1712.04144.

[38] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised Generative Modeling Using Matrix Product States. 2017. URL http://arxiv.org/abs/1709.01662.

[39] Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blázquez García, Gang Su, and Maciej Lewenstein. Machine Learning by Two-Dimensional Hierarchical Tensor Networks: A Quantum Information Theoretic Perspective on Deep Architectures. *arXiv*, 2017. URL http://arxiv.org/abs/1710.04833.

[40] E. M. Stoudenmire. Learning Relevant Features of Data with Multi-scale Tensor Networks. 2017. URL http://arxiv.org/abs/1801.00315.

[41] Raphael Bailly. Quadratic weighted automata:spectral algorithm and likelihood maximization. In Chun-Nan Hsu and Wee Sun Lee, editors, *Proceedings of the Asian Conference on Machine Learning*, volume 20 of *Proceedings of Machine Learning Research*, pages 147–163, South Garden Hotels and Resorts, Taoyuan, Taiwain, 14–15 Nov 2011. PMLR. URL http://proceedings.mlr.press/v20/bailly11.html.

[42] Ming-Jie Zhao and Herbert Jaeger. Norm-observable operator models. *Neural Computation*, 22(7):1927–1959, 2010. doi: 10.1162/neco.2010.03-09-983. URL https://doi.org/10.1162/neco.2010.03-09-983. PMID: 20141473.

[43] W. Huggins, P. Patel, K. B. Whaley, and E. Miles Stoudenmire. Towards Quantum Machine Learning with Tensor Networks. *ArXiv e-prints*, March 2018.

[44] Yoav Levine, David Yakira, Nadav Cohen, and Amnon Shashua. Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design. 2017. URL http://arxiv.org/abs/1704.01552.

[45] A. Cichocki, A-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives. *arXiv*, 2017. doi: 10.1561/2200000067. URL http://arxiv.org/abs/1708.09165.

[46] J.-G. Liu and L. Wang. Differentiable Learning of Quantum Circuit Born Machine. *ArXiv e-prints*, April 2018.

[47] H. Situ, Z. He, L. Li, and S. Zheng. Adversarial training of quantum Born machine. *ArXiv e-prints*, July 2018.

[48] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[49] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. *arXiv*, 2016. URL http://arxiv.org/abs/1611.02731.

[50] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355: 602, 2017.

[51] Stephen R. Clark. Unifying Neural-network Quantum States and Correlator Product States via Tensor Networks. 2017. URL http://arxiv.org/abs/1710.03545.

[52] Ivan Glasser, Nicola Pancotti, Moritz August, Ivan D. Rodriguez, and J. Ignacio Cirac. Neural Networks Quantum States, String-Bond States and chiral topological states. *Phys. Rev. X*, 8:11006, 2017. doi: 10.1103/PhysRevX.8.011006. URL http://arxiv.org/abs/1710.04045.

[53] Dong Ling Deng, Xiaopeng Li, and S. Das Sarma. Quantum entanglement in neural network states. *Phys. Rev. X*, 7:1–17, 2017. doi: 10.1103/PhysRevX.7.021021.

[54] Xun Gao and Lu-Ming Duan. Efficient Representation of Quantum Many-body States with Deep Neural Networks. *Nat. Commun.*, pages 1–5, 2017. doi: 10.1038/s41467-017-00705-2. URL http://arxiv.org/abs/1701.05039{%}0Ahttp://dx.doi.org/10.1038/s41467-017-00705-2.

[55] Raphael Kaubruegger, Lorenzo Pastori, and Jan Carl Budich. Chiral Topological Phases from Artificial Neural Networks. *arXiv*, 2017. URL http://arxiv.org/abs/1710.04713.

[56] Zi Cai and Jinguo Liu. Approximating quantum many-body wave-functions using artificial neural networks. 035116:1–8, 2017. doi: 10.1103/PhysRevB.97.035116. URL http://arxiv.org/abs/1704.05148.

[57] J. Han, L. Zhang, H. Wang, and W. E. Solving Many-Electron Schro2dinger Equation Using Deep Neural Networks. *ArXiv e-prints*.

[58] D. Wu, L. Wang, and P. Zhang. Solving Statistical Mechanics using Variational Autoregressive Networks. *ArXiv e-prints*, September 2018.

[59] Cédric Bény. Deep learning and the renormalization group. *arXiv*, 2013. URL http://arxiv.org/abs/1301.3124.

[60] Pankaj Mehta and David J. Schwab. An exact mapping between the Variational Renormalization Group and Deep Learning. *arXiv*, 2014. URL http://arxiv.org/abs/1410.3831.

[61] Serena Bradde and William Bialek. PCA Meets RG. *J. Stat. Phys.*, 167:462–475, 2017. doi: 10.1007/s10955-017-1770-6.

[62] Maciej Koch-Janusz and Zohar Ringel. Mutual Information, Neural Networks and the Renormalization Group. *arXiv*, 2017. URL http://arxiv.org/abs/1704.06279.

[63] Shuo-Hui Li and Lei Wang. Neural Network Renormalization Group. *arXiv*, 2018. URL http://arxiv.org/abs/1802.02840.

[64] Yi-Zhuang You, Zhao Yang, and Xiao-Liang Qi. Machine learning spatial geometry from entanglement features. *Phys. Rev. B*, 97:045153, Jan 2018. doi: 10.1103/PhysRevB.97.045153. URL https://link.aps.org/doi/10.1103/PhysRevB.97.045153.

[65] C. E. Rasmussen. Gaussian Processes to Speed up Hybrid Monte Carlo for Expensive Bayesian Integrals. *Bayesian Stat. 7*, pages 651–659, 2003. URL http://www.is.tuebingen.mpg.de/fileadmin/user{_}upload/files/publications/pdf2080.pdf.

[66] Simon Duane, A D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Phys. Lett. B*, 195:216–222, 1987. doi: https://doi.org/10.1016/0370-2693(87)91197-X. URL http://www.sciencedirect.com/science/article/pii/037026938791197X.

[67] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning Monte Carlo method. *Phys. Rev. B*, 95:1–5, 2017. doi: 10.1103/PhysRevB.95.041101.

[68] Li Huang, Yi Feng Yang, and Lei Wang. Recommender engine for continuous-time quantum Monte Carlo methods. *Phys. Rev. E*, 95:031301(R), 2017. doi: 10.1103/PhysRevE.95.031301.

[69] Li Huang and Lei Wang. Accelerated Monte Carlo simulations with restricted Boltzmann machines. *Phys. Rev. B*, 95:035105, 2017. doi: 10.1103/PhysRevB.95.035105.

[70] Junwei Liu, Huitao Shen, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning Monte Carlo method and cumulative update in fermion systems. *Phys. Rev. B*, 95:241104, 2017. doi: 10.1103/PhysRevB.95.241104.

[71] Faming Liang, Chuanhai Liu, and Raymond J Carroll. *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*. Wiley, 2011.

[72] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-NICE-MC: Adversarial Training for MCMC. *arXiv*, 2017. URL http://arxiv.org/abs/1706.07561.

[73] Daniel Levy, Matthew D. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with Neural Networks. *arXiv*, 2017. URL http://arxiv.org/abs/1711.09268.

[74] Marco F. Cusumano-Towner and Vikash K. Mansinghka. Using probabilistic programs as proposals. *arXiv*, 2018. URL http://arxiv.org/abs/1801.03612.

[75] Troels Arnfred Bojesen. Policy Guided Monte Carlo: Reinforcement Learning Markov Chain Dynamics. *arXiv*, 2018. URL http://arxiv.org/abs/1808.09095.

[76] Lei Wang. Can Boltzmann Machines Discover Cluster Updates ? *arXiv*, 2017. doi: 10.1103/PhysRevE.96.051301. URL http://arxiv.org/abs/1702.08586.

[77] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv*, 2016. doi: 10.1021/acscentsci.7b00572. URL http://arxiv.org/abs/1610.02415.

[78] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[79] Giacomo Torlai and Roger G. Melko. Neural Decoder for Topological Codes. *Phys. Rev. Lett.*, 119:030501, 2017. doi: 10.1103/PhysRevLett.119.030501.

[80] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Many-body quantum state tomography with neural networks. *arXiv*, 2017. URL http://arxiv.org/abs/1703.05334.

[81] Dong-Ling Deng. Machine learning detection of bell nonlocality in quantum many-body systems. *Phys. Rev. Lett.*, 120:240402, Jun 2018. doi: 10.1103/PhysRevLett.120.240402. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.240402.

[82] Scott Aaronson. Read the fine print. *Nat. Phys.*, 11:291–293, 2015. doi: 10.1038/nphys3272. URL http://dx.doi.org/10.1038/nphys3272.

[83] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017. doi: 10.1038/nature23474. URL http://dx.doi.org/10.1038/nature23474.

[84] Xun Gao, Zhengyu Zhang, and Luming Duan. An efficient quantum algorithm for generative machine learning. *arXiv*, 2017. URL http://arxiv.org/abs/1711.02038.

[85] Marcello Benedetti, Delfina Garcia-Pintos, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *arXiv*, 2018. URL http://arxiv.org/abs/1801.07686.

[86] J. Zeng, Y. Wu, J.-G. Liu, L. Wang, and J. Hu. Learning and Inference on Generative Adversarial Quantum Circuits. *ArXiv e-prints*, August 2018.